DSP 编程的几个关键问题

DSP 芯片凭其优异的性能在高速计算领域有着巨大的应用前景。但其应用所涉及的知识非常庞杂。本文以 TI 公司 320C54X 系列为蓝本进行提纯，所有认识都是笔者在实际工作中亲手实践所得。当程序调不通不知该从何处下手时，此文也许会有所帮助。这些关键点有些是 TMS320C5409 所触有而有些是与 DSP 所共有的。

1 McBSP(Multichannel Buffered Serial Port)串口利用 DMA 中的多帧（Multi-Frame）方式通信的中断处理

在实际通信应用中，一个突发之后，程序必须为下一个突发作准备。因此一般采用串口的 DMA 多帧方式但在串口以 DMA 方式传输数据时却有一些问题要讨论。首先 DMA 的传输同步事件应设 McBSP 的传输事件即 XEVT，这样一字节传输后会自动准备另一字节（McBSP 的 READY 上升沿触发 DMA 传输）。中断发生时意味着一个块已传完，这时 DMA 的使能自动关闭，McBSP 的 READY 将一直保持高状态。但是在下一次突发传输直接使能 DMA 时却启动不了传输（相信会有许多我遇到此类问题）。这是因为无法产生 McBSP 触发启动所需的 READY 上升沿。解决办法是在中断程序中先关闭 McBSP 的发送，使 READY=0，随后在程序中发送使能 DMA，再打开 McBSP 的发送即可。如先打开 McBSP 的发送后打开 DMA，也是不会工作的。因为 McBSP 的 READY 已经由 0 变到 1 了，无法再产生 READY 上升沿。

2 关闭 DMA 与关闭 McBSP 的区别

在通信领域，为了充分利用 DSP 的片上外设资源，常常利用 DMA 把从串口来的数据或要发的数据放入缓冲区，再处理。对 DMA 而言，只要其在数据缓冲区的指针指向了中断应发生的位置，就产生中断。但此时最后一个数据只是进入了 McBSP 而并未真正发出去，所以在传送结束的中断程序中只能关闭 DMA 不能关闭 McBSP。因为此时 McBSP 的发寄存器 DXR 中还有一个字没有发出。

3 McBSP 串口配置的关键时序

主要是寄存器 SPCR2 的配置：在保持 RRST、XRST、FRST 各位为 0 的前提下，配置好其它串口控制寄存器。等待至少 2 个 CLKR/T 时钟以确保 DSP 内部的同步。

（1）可以向 DXR 装载数据或使能 DMA。

（2）使能 GRST（GRST=1）（如果需要 DSP 内部产生采样时钟）。

（3）使能 RRST 或 XRST，注意此时要保证 SPCR 中仅有此一位发生改变。

（4）使能 FRST（FRST=1）（如果需要 DSP 内部产生帧同步）。

（5）等待 2 个 R/T CLK 时钟周期后，收或发端便会有效。

4 汇编语言程序中的变量

汇编语言程序中的公用变量应在文件中定义，如.def carry。汇编语言程序中使用的局部变量不需定义，可直接声明，例如 trn_num .word 00h。如果在两个 asm 文件中有两个都没有定义的同名变量，则编译程序会认为分他们不是同一变量。在汇编程序的开头应有.mmregs 宏语句。它一方面表示对默认定义的确认（ah,bh,trn 等），另一方面可以对所用寄存器重新定义。如：

```
.mmregs

DMPREC .set 54h ;定义 DMA 优先和使能寄存器地址在 54h

DMSA .set 55h

DMSDN .set 57h


DXR10 .set 23h ;定义串口 1 的发送寄存器地睛在 23h
```

5 ST1 寄存器中 CPL 位的影响

CPL 位是编译模式控制位，它表示在相对直接寻址时采用哪种指针。当 CPL=0 时，使用页指针 DP；当 CPL=1 时，使用堆栈指针 SP。实际使用中二者没有什么差别，但使用 SP 寻址的程序更易读。在程序中经常使用 CPL=1。

6 指令的歧义

6.1 比较下面指令

STLM B，AR4 ；把 bl 内容送入寄存器 AR4 （×）

STLM B,*AR4 ；把 bl 内容送入寄存器 AR4 （√）

前者实际执行的是把 bl 内容送入一个系统用的缓冲区，后者也可用：

MVDM BL，AR4 ；把 bl 内容送入寄存器 AR4 （√）

其他易导致歧义的语句还有：

LD AR5，A ；把 AR5 的内容送入寄存器 A （×）

LDM AR5，A ；把 AR5 的内容送入寄存器 A （√）

ANDM #0x107e,AR4; 把#107e 加到寄存器 AR4 （×）

ANDN #0x107e，*AR4;把#107e 加到寄存器 AR4 （√）

仅对某些寄存器有效的指令：

MVDD * AR2+，*AR3+ ；把以 AR2 为地址的内容拷入 AR3 的地址中

此类指令用作数据块搬移特别有效，但仅对 AR2、AR3、AR4、AR5 有效。

易错语句中对程序运行危害最大的是：

ST #0，*（bsp0_out_sign） ;bsp0_out_sign 是一个变量名（√）

STM #0，bsp0_out_sign ；此语句被编译为 STM #0,PMST 或 STM #0，IMR （×）

这种语句会导致程序运行中的随机故障，且极难发现。

6.2 流水冲突

分析以下程序：

```
STM to_dce_buff,AR4

LDM AR4,B

ADD A,B ;B=AR4+AL

MVDM BL,AR4 ;AR4=to-dce-buff+AL
```

实际上，上段程序得不到 AR4=to-dce-buff+AL 的结果。这是因为 DSP 一般采用深度为 3～6 级的流水结构，产生了无法解决的冲突，所以它不能被正确执行。解决的办法是在赋值和引用之间插入一条或几条其他的指令，或 NOP 语句即可。

7 汇编与 C 语言混合编程的关键问题

7.1 C 程序变量与汇编程序变量的共用

为了使程序更易于接口和维护，可以在汇编程序中引用与 C 程序共享的变量：

```
.ref_to_dce_num,_to_dte_num,_to_dce_buff,_to_dte_buff
```

在汇编程序中引用而在 C 程序可直接定义的变量：

```
unsigned char to_dte_buff[BUFF_SIZE]; //DSP 发向 PC 机的数据

int to_dte_num; //缓冲区中存放的有效字节数

int to_dte_store:  //缓冲区的存放指针

int to_dte_read; //缓冲区的读取指针
```

这样经过链接就可完成对应。

7.2 程序入口问题

在 C 程序中，程序的入口是 main（）函数。而在汇编程序中其入口由*.cmd 文件中的命令决定，如：-emain_start；程序入口地址为 main_start。这样，混合汇编出来的程序得不到正确结果。因为 C 到 ASM 的汇编有默认的入口 c-int00，从这开始的一段程序为 C 程序的运行做准备工作。这些工作包括初始化变量、设置栈指针等，相当于系统壳不能耐跨越。这时可在*.cmd 文件中去掉语句：-e main_start。如仍想执行某些汇编程序，可以 C 函数的形式执行，如：

```
main_start();  //其中含有其他汇编程序
```

但前提是在汇编程序中把_main_start 作为首地址，程序以 rete 结尾（作为可调用的函数）的程序段，并在汇编程序中引用_main_start，即.ref _main_start。

7.3 移位问题

在 C 语言中把变量设为 char 型时，它是 8 位的，但在 DSP 汇编中此变量仍被作为 16 位处理。所以会出现在 C 程序中的移位结果与汇编程序移位结果不同的问题。解决的办法是在 C 程序中，把移位结果再用 0X00FF 去“与”一下即可。

7.4 堆栈问题

在汇编程序中对堆栈的依赖很小，但在 C 程序中分配局部变量、变量初始化、传递函数变量、保存函数返回地址、保护临时结果功能都是靠堆栈完成。而 C 编译器无法检查程序运行时堆栈能否溢出。所以应尽量多给堆栈分配空间。C 编译器的默认大小为 1KB。在程序不正常跑飞时应注意检查是否堆栈溢出。

7.5 程序跑飞问题

编译后的 C 程序跑飞一般是对不存在的存储区访问造成的。首先要查.MAP 文件并与 memery map 图对比，看是否超出范围。如果在有中断的程序中跑飞，应重点查在中断程序中是否对所用到的寄存器进行了压栈保护。如果在中断程序中调用了 C 程序，则要查汇编后的 C 程序中是否用到了没有被保护的寄存器并提供保护（在 C 程序的编译中是不对 A、B 等寄存器进行保护的）。

8 命令文件的编写

在编辑*.cmd 文件时编译连接器默认：page 0 就是 ROM 区，page 1 就是 RAM 区。下列段必须放在 ROM 区。

.text load=PROG PAGE 0 ;程序段

.const load=data PAGE 0 ;常数段

.cinit load=data PAGE 0 ;初始化段

.switch load=data PAGE 0 ;switch 指令常数表

值得注意的是尽量不要用 FILL 选项,一旦进行填充会使生成的.out 文件增大甚至超过内部的存储空间而无法 Bootload。

9 BOOtload 问题

一般都采用从 EPROM 引导，但通常很费脑筋。下面介绍一下可为 54X 系列 DSP 内部引导程序识别的 EPROM 存储结构，如表 1 所示。

EPROM 内容

地 址

08AAh 或 10AAh

SWWSR(等待状态产生寄存器）值 16

BSCR（页切换控制寄存器）值 16

人口点 XPC（外部存储器映射寄存器）值 7

人口点 PC（程序地址寄存器）值 16

第一块的大小 16

第一块的人口点 XPC（外部存储器映射寄存器）值 7

第一块的人口点 PC（程序地址寄存器）值 16

代码（1）16

……

代码（N）16

最后一块的大小 16

最后一块的人口点 XPC（外部存储器映射寄存器）值 7

最后一块的人口点 PC（程序地址寄存器）值 16

代码（1）16

……

代码（N）16

0000h(标志引导表结束)

……

……

……

EPROM 的启始地址（如 8000h）

首地址

FFFFh

假使已经生成了*.out 文件，生成时必须带有芯片，此处为 MS320VC5409，版本参数如：asm500 init_54x-v548）。

.hex 文件与 EPROM 的存储空间相对应，其生成的参数由.cmd 文件决定。下面以实例介绍.cmd 文件中的参数编写及意义。

cdpd.out ； 将 cdpd.out 文件转换成.hex 文件

-SWWSR 7fffh ； 将外部设备的等待时间设为 7 个等待状态

-BSCR 0f800h ； 设置 4K 为一页，页面切换时插入 1 个等待状态

-o cdpd.hex ； 转换成 cdpd.hex 文件

-i ;intel 格式

-boot ;把所有的程序块装入 EPROM

-bootorg 8000h ； 从 EPROM 存储器的 8000h 处开始写入程序内容

-memwidth 8 ;系统数据宽度转为 8 位，以避免生成 2 个文件

-romwidth 8 ;EPROM 数据宽度为 8 位

-e 0840h ;从 8040h 开始执行程序

-map wfcdpd.mxp ； 生成 EPROM 存储器占用映射

这时生成的 cdpd.hex 可以直接写入 EPROM。需要说明 320C5409 的外部 RAM 范围从 8000h～FFFFh，所以设首地址为 8000h。但是对 C54x 系列而言，其转换有个 BUG，即它总是不能在 0XFFFF 处写入从外部 EPROM 存储器装载的开始地址，只好自己填入。对本例而言在 0XFFFE 处写 0X80，在 0XFFFF 处写 0X00。

相信对你有帮助的：

CCS 编程入门（全）

CCS3.3 安装常见问题集锦 1

CCS3.3 安装常见问题集锦 2

用 DSP 编程 spwm 波形的产生